COM S 687 Introduction to Cryptography	September 28, 2006

Lecture 11: The Goldreich-Levin Theorem

Instructor: Rafael Pass — *Scribe: Krishnaprasad Vikram*

Hard-Core Bits

Definition: A predicate $b : \{0,1\}^\gamma \rightarrow \{0,1\}$ is hardcore for a function f if

(a) b is efficiently computable

(b) $\forall p.p.t.\ \mathcal{A}, \exists$ a negligible polynomial ϵ s.t.
$\forall k\ Pr[X \leftarrow \{0,1\}^k : \mathcal{A}(1^k, f(X) = b(X))] \leq \frac{1}{2} + \epsilon(k)$

Intuitively, a hardcore bit (described as a function b) is efficiently computable given an input x, but is *hard* to compute given only $f(x)$. In other words, f *hides* the bit b. This definition can be trivially extended to collections of one-way functions.

Construction of a PRG[1]

Using the idea of hardcore bits, and assuming the existence of a one-way permutation f we constructed in the previous lecture a PRG $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ given by

$$G(s) \quad = \quad f(s)||b(s)$$

where f is a one-way *permutation* and b is a hard core bit for f, and $||$ is the string concatenation operator. Intuitively, this is a PRG given that it passes the *next-bit test* since it would be hard to compute the $n+1^{th}$ bit $b(s)$ given the first n bits $f(s)$. However, this directly does not hold true of a OWF (Why? We might be proving that in an upcoming homework).

However, there is a theorem that says "$\exists$ of a OWF $\Leftrightarrow$ $\exists$ of a Pseudo-random number generator"[3]. We'll prove the (supposedly easy) $\Leftarrow$ direction in one of the homeworks.

Now, we'll show in class that "$\exists$ of a OWP $\Rightarrow$ $\exists$ of a PRG". Note that it is an open problem to prove that any OWF or a OWP has a hard core bit. What we'll show is that every OWF (or OWP) can be transformed into a new OWF (respectively OWP) that has a hard-core bit.

One possibility of a hardcore bit is a parity function, but that might be easy to compute given $f(x)$. We'll try something more sophisticated.

Theorem

Let f be a OWF. Then $f'(X,r) = f(X), r$ (where $|X| = |r|$) is a OWF and $b(X,r) = \langle X, r\rangle_2 = \Sigma X_i r_i \bmod 2$ (inner product mod 2) is a hardcore predicate for f.

Here r essentially tells us which bits to take parity of. Note that f' is a OWP if f is a OWP.

Proof. (by *reductio ad absurdum*) We show that if $\mathcal{A}$, given $f'(X,r)$ can compute $b(X,r)$ w.p. significantly better than $1/2 \Rightarrow \exists$ p.p.t. $\mathcal{B}$ that inverts f.

We'll do the proof in three steps. In the first step, we consider a very oversimplified case and prove the theorem for that case. In the next step, we take a less simplified case and finally we take the general case.

In the very **oversimplified** case, we assume $\mathcal{A}$ always computes b correctly. And so, we can construct a f' with an r such that the first bit is 1 and the other bits are 0. In such a case $\mathcal{A}$ would return the first bit of X. Similarly we can set the second bits of r to be 1 to obtain the second bit of X. Thus, we have $\mathcal{B}$ given by

$\mathcal{B}(y)$: Let $X_i = \mathcal{A}(y, e_i)$ where $e_i = 000\ldots1\ldots000$ where the 1 is on the i^{th} position.

-Output $X_1, X_2, \ldots, X_n$

This works, since $\langle X, e_i\rangle_2 = X_i$

Now, in the **less simplified** case, we assume that $\mathcal{A}$, when given random $y = f(X)$ and random r, computes $b(X,r)$ w.p. $\frac{3}{4} + \epsilon$, ($\epsilon = \frac{1}{poly(n)}$, n is the length of X).

Intuition: we want the attacker to compute b with a fixed X and a varying r so that given enough observations, X can be computed eventually. The trick is to find the set of good X, for which this will work.

As an attempt to find such X, let $S = \{X | \Pr[\mathcal{A}(f(X), r) = b(X,r)] > \frac{3}{4} + \frac{\epsilon}{2}\}$. It can be shown that $|S| > \epsilon/2$.

A simple attack with various e_i might not work here. More rerandomization is required. Idea: Use linearity of $\langle a, b\rangle$.

Useful relevant fact: $\langle a, b \oplus c\rangle = \langle a, b\rangle \oplus \langle a, c\rangle \ mod\ 2$

Proof.

$$\begin{aligned}
\langle a, b \oplus c\rangle &= \Sigma a_i(b_i + c_i) \\
&= \Sigma a_i b_i + \Sigma a_i c_i \\
&= \langle a, b\rangle + \langle a, c\rangle \ mod\ 2
\end{aligned}$$

Attacker asks: $\langle X, r\rangle, \langle X, r + e_1\rangle$

and then XOR both to get $\langle X, e_1 \rangle$ without ever asking for e_1.

And so, $\mathcal{B}$ inverts f as follows: $\mathcal{B}(y)$:

For $i = 1$ to n

1. Pick random r in $\{0,1\}^n$
2. Let $r' = e_i \oplus r$
3. Compute guess for X_i as $\mathcal{A}(y, r) \oplus \mathcal{A}(y, r')$
4. Repeat $poly(1/\epsilon)$ times and let X_i be majority of guesses.

Finally output $X_1, \ldots, X_n$.

If we assume e_1 and $r + e_1$ as independent, the proof works fine. However, they are not independent. The proof is still OK though, as can be seen using the union bound:

The proof works because:

- w.p. $\frac{1}{4} - \frac{\epsilon}{2}$ $\mathcal{A}(y, r) \neq b(X, r)$
- w.p. $\frac{1}{4} - \frac{\epsilon}{2}$ $\mathcal{A}(y, r') \neq b(X, r)$
- by union bound w.p. $\frac{1}{2}$ both answers of $\mathcal{A}$ are OK.
- Since $\langle y, r \rangle + \langle y, r' \rangle = \langle y, r \oplus r' \rangle = \langle y, e_i \rangle$, each guess is correct w.p. $\frac{1}{2} + \epsilon$
- Since samples are independent, using Chernoff Bound it can be shown that every bit is OK w.h.p.

Now, to the **general** case. Here, we assume that $\mathcal{A}$, given random $y = f(X)$, random r computes $b(X, r)$ w.p. $\frac{1}{2} + \epsilon$ $\left(\epsilon = \frac{1}{poly(n)}\right)$

Let $S = \{X | \Pr[\mathcal{A}(f(X), r) = b(X, r)] > \frac{1}{2} + \frac{\epsilon}{2}\}$. It again follows that $|S| > \frac{\epsilon}{2}$.

Assume set access to oracle C that given $f(X)$ gives us samples

$$\begin{array}{l} \langle X, r_1 \rangle, r_1 \\ \qquad \vdots \qquad\qquad \text{(where } r_1, \ldots, r_n \text{ are independent and random)} \\ \langle X, r_n \rangle, r_n \end{array}$$

We now recall Homework 1, where given an algorithm that computes a correct bit value w.p. greater than $\frac{1}{2} + \epsilon$, we can run it multiple times and take the majority result, thereby computing the bit w.p. as close to 1 as desired.

From here on, the idea is to eliminate C from the constructed machine step by step, so that we don't need an oracle in the final machine $\mathcal{B}$.

Consider the following $\mathcal{B}(y)$:

For $i = 1$ to n

1. $C(y) \rightarrow (b_1, r_1), \ldots, (b_m, r_m)$
2. Let $r'_j = e_i \oplus r_j$
3. Compute $g_j = b_j \oplus \mathcal{A}(y, r')$
4. Let $X_i = majority(g_1, \ldots, g_m)$

Output $X_1, \ldots, X_m$

Each guess g_i is correct w.p. $\frac{1}{2} + \frac{\epsilon}{2} = \frac{1}{2} + \epsilon'$. As in HW1, by Chernoff bound, an x_i is wrong w.p. $\leq 2^{-\epsilon'^2 m}$ (was $2^{-4\epsilon^2 m}$ in the HW). If $m >> \frac{1}{\epsilon'^2}$, we are OK.

Now, we assume that C gives us samples $\langle X, r_1 \rangle, r_1; \ldots; \langle X, r_n \rangle, r_n$ which are random but only **pairwise independent**. Again, using results from HW1, by Chebyshev's theorem, each X_i is wrong w.p. $\leq \frac{1-4\epsilon'^2}{4m\epsilon'^2} \leq \frac{1}{m\epsilon'^2}$ (ignoring constants).

By union bound, any of the X_i is wrong w.p. $\leq \frac{n}{m\epsilon'^2} \leq \frac{1}{2}$, when $m \geq \frac{2n}{\epsilon'^2}$. Therefore, as long as we have polynomially many samples (precisely $\frac{2n}{\epsilon'^2}$ pairwise independent samples), we'd be done.

The question now is: How do we get pairwise independent samples? So, our initial attempt to remove C would be to pick $r_1, \ldots, r_m$ on random and guess $b_1, \ldots, b_m$ randomly. However, b_i would be correct only w.p. 2^{-m}.

A better attempt is to pick $log(m)$ samples $s_1, \ldots, s_{log(m)}$ and guessing $b'_1, \ldots, b'_{log(m)}$ randomly. Here the guess is correct with probability $1/m$.

Now, generate $r_1, r_2, \ldots, r_{m-1}$ as all possible sums (mod 2) of subsets of $s_1, \ldots, s_{log(m)}$, and $b_1, b_2, \ldots, b_m$ as the corresponding subsets of b'_i. Mathematically

$$\begin{aligned} r_i &= \sum_{j \in I_i} s_j \quad j \in I \text{ iff } i_j = 1 \\ b_i &= \sum_{j \in I_i} b'_j \end{aligned}$$

In HW1, we showed that these r_i are pairwise independent samples. Yet w.p. $1/m$, all guesses for $b'_1, \ldots, b'_{log(m)}$ are correct, which means that $b_1, \ldots, b_{m-1}$ are also correct.

Thus, for a fraction of ϵ' of X' it holds that w.p. $1/m$ we invert w.p. $1/2$. That is $\mathcal{B}(y)$ inverts w.p.

$$\frac{\epsilon'}{2m} = \frac{\epsilon'^3}{4n} = \frac{(\epsilon/2)^3}{4n} \quad (m = \frac{2n}{\epsilon^2})$$

which contradicts the (strong) one-way-ness of f.

Yao proved that if OWF exists, then there exists OWF with hard core bits. But this construction is due to Goldreich and Levin[2] and by Charles Rackoff[**?**].

References

[1] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[2] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, New York, NY, USA, 1989. ACM Press.

[3] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudo-random generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.